

1/15

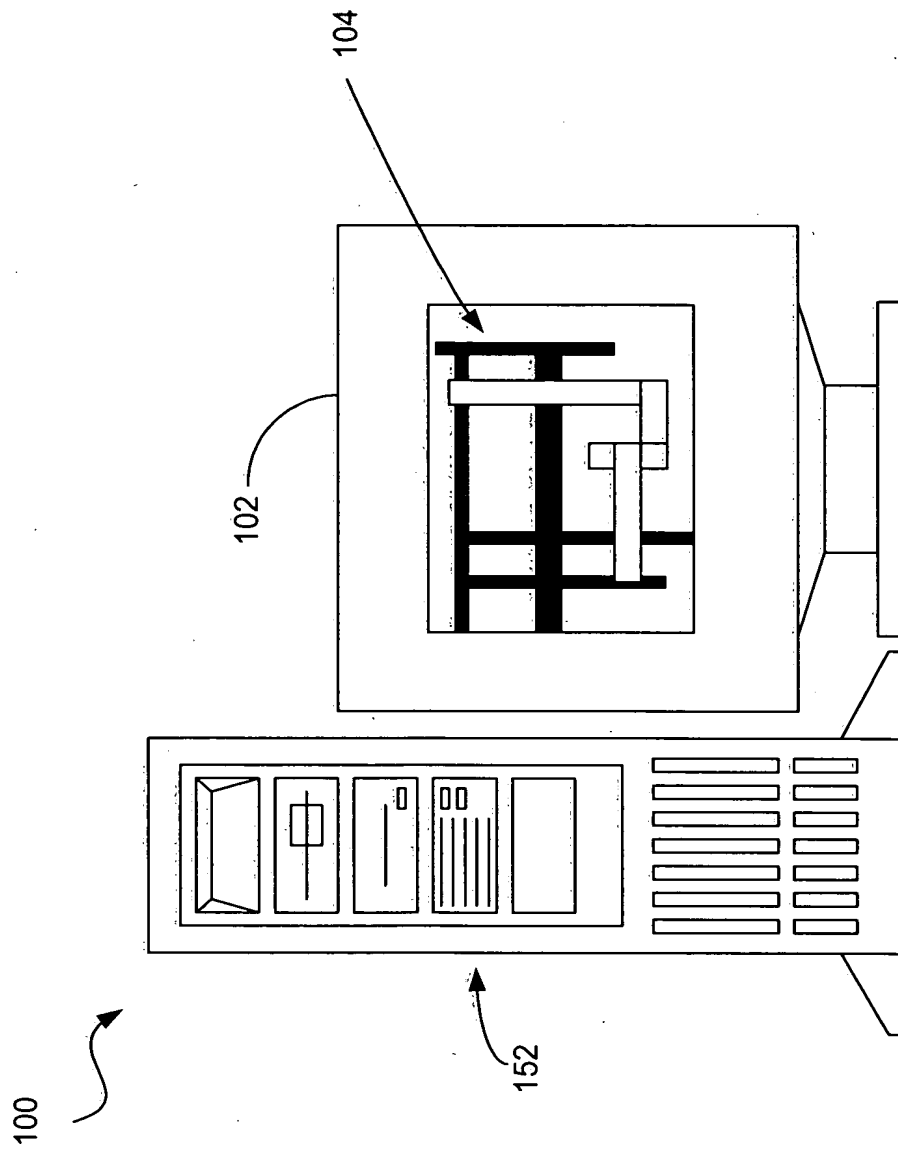


FIG. 1

2/15

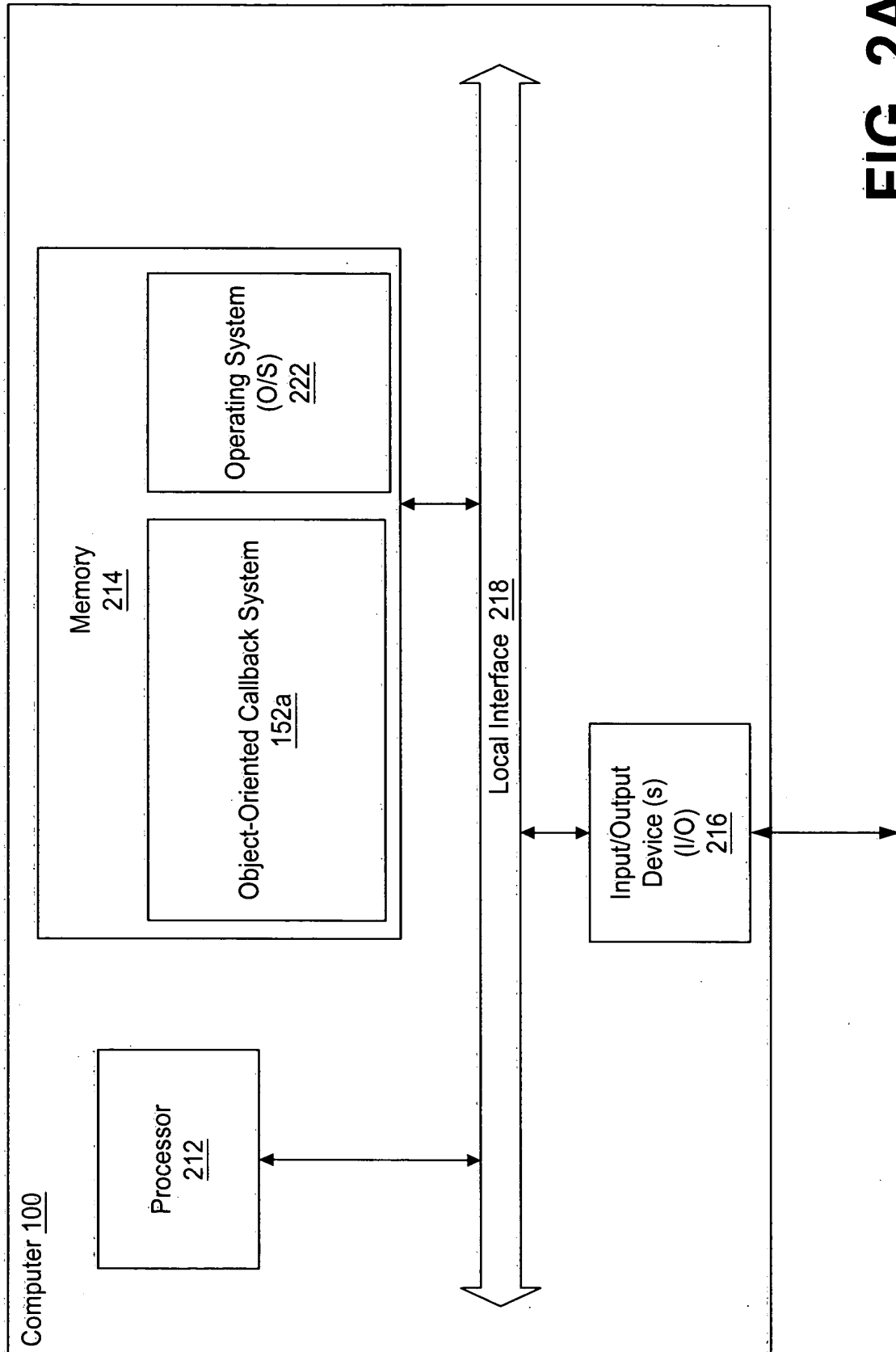
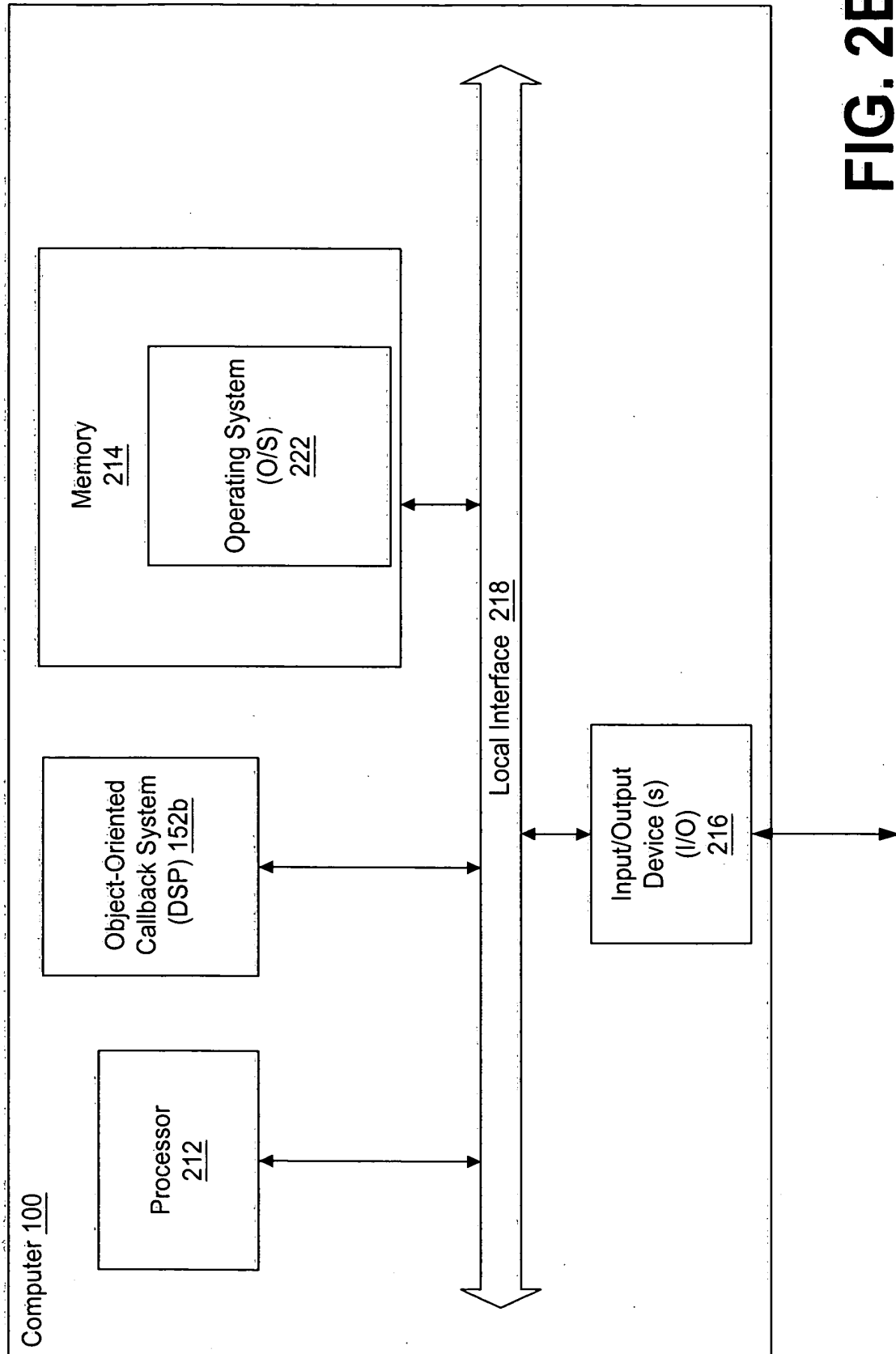


FIG. 2A

3/15



**FIG. 2B**

4/15

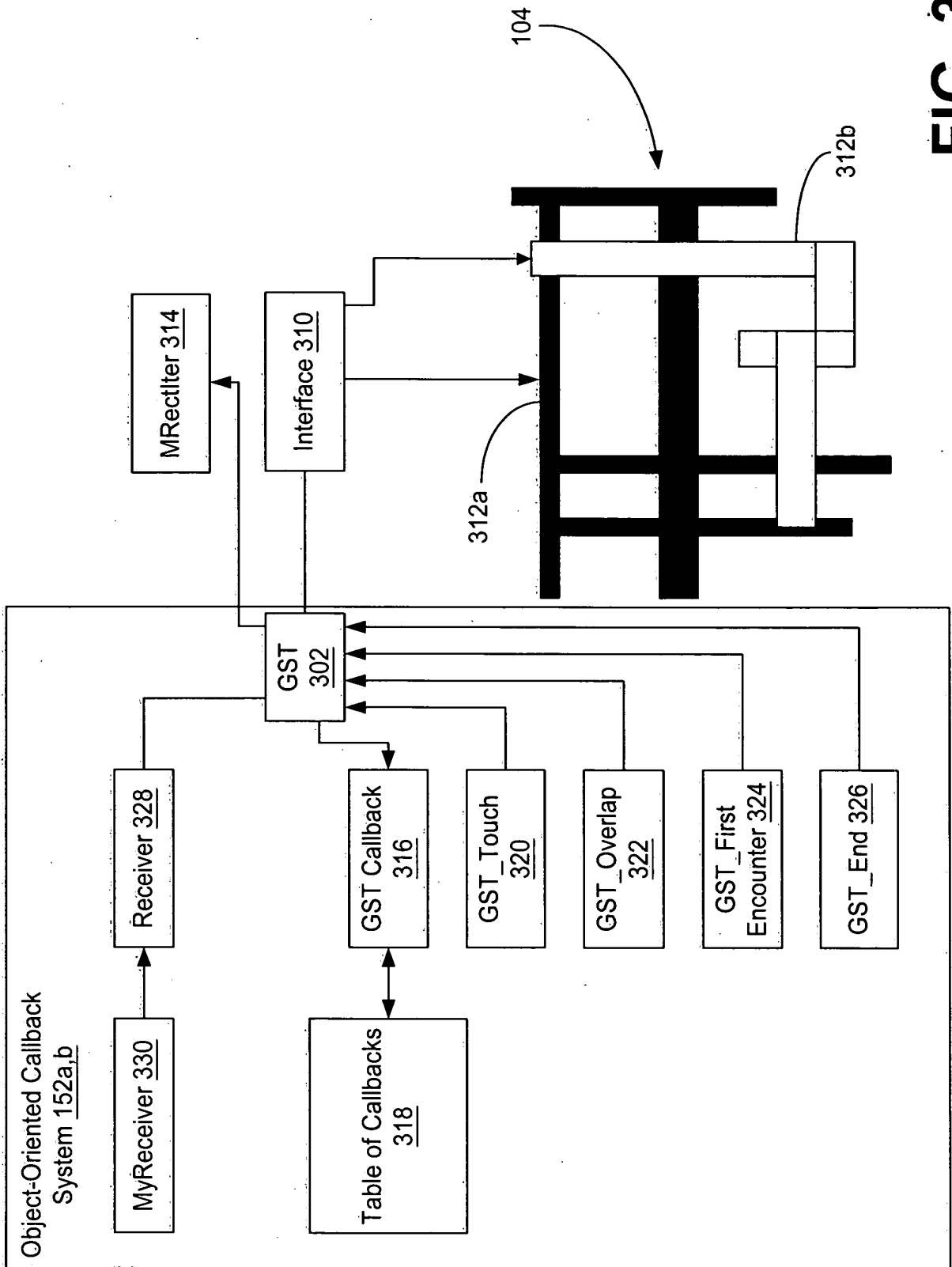
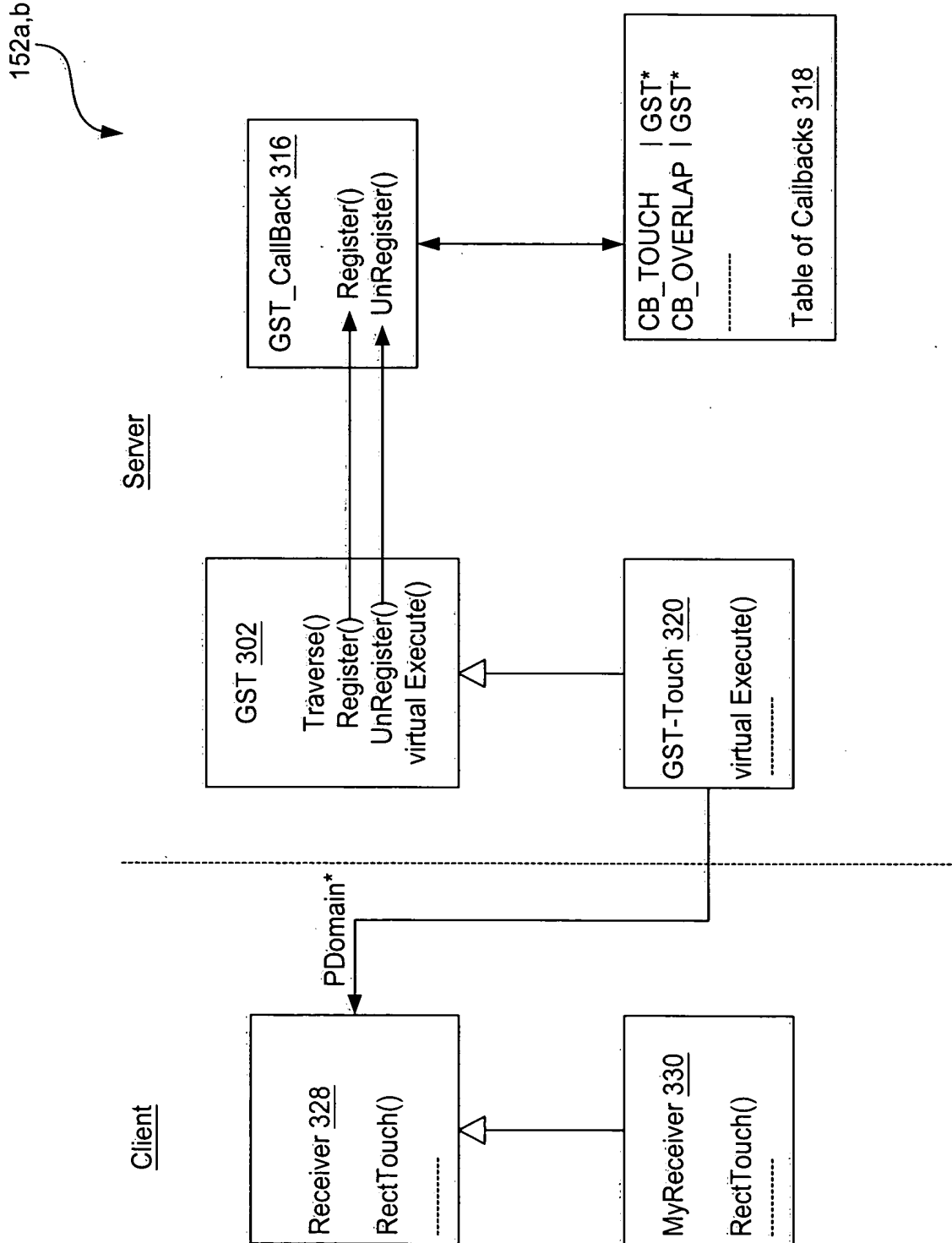


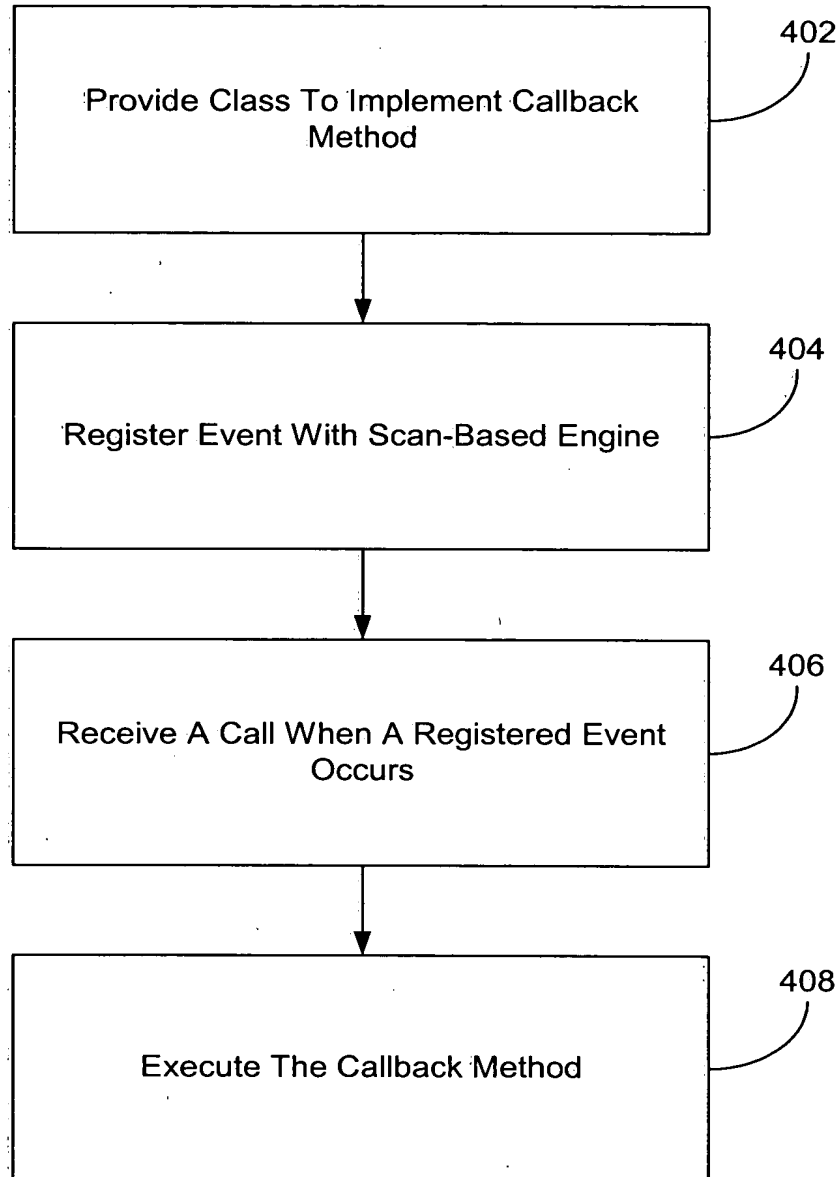
FIG. 3



**FIG. 4A**

6/15

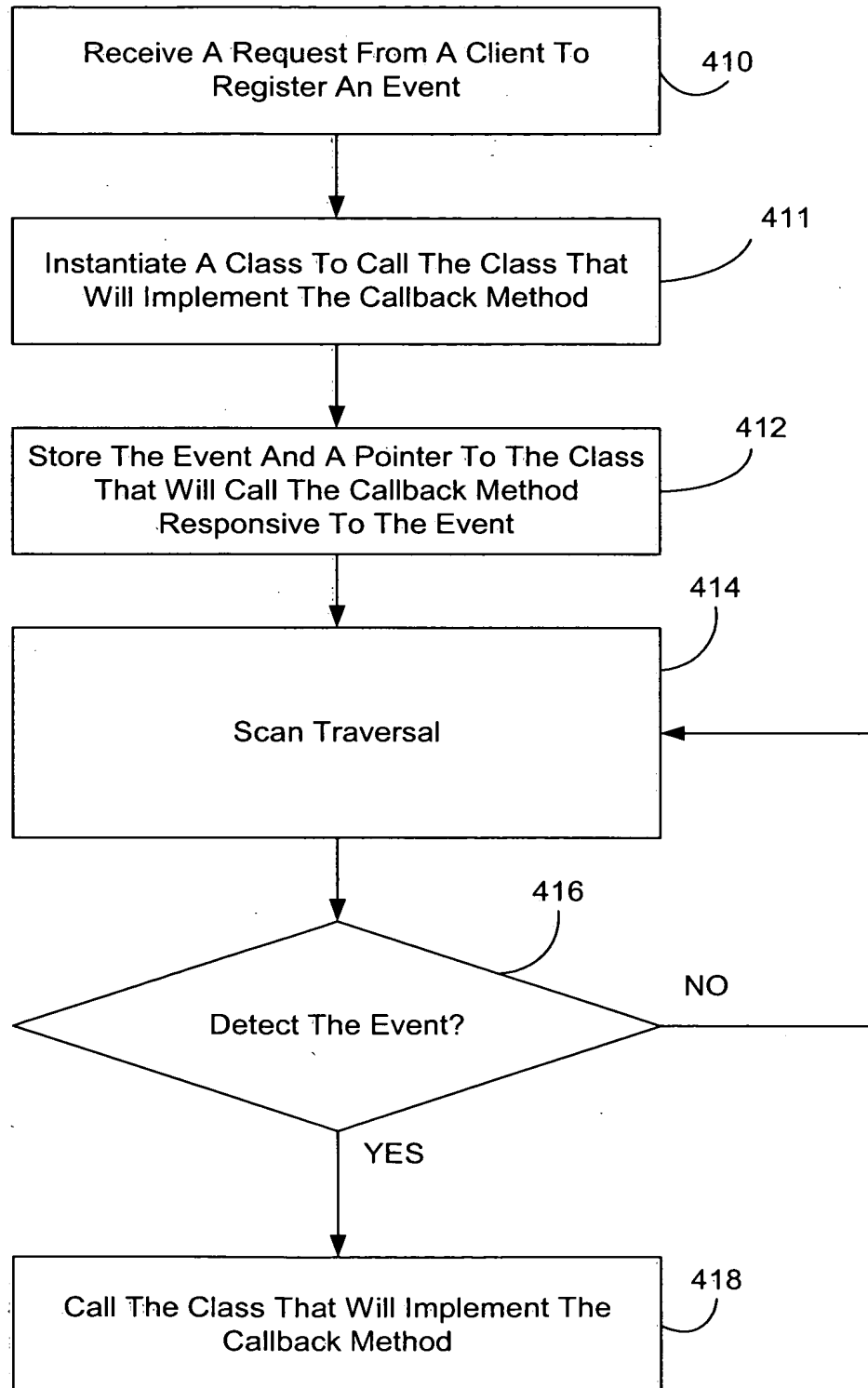
Client Perspective



**FIG. 4B**

7/15

Server Perspective



**FIG. 4C**

```

506 {
507     public:
508         virtual void RectFirstEncounter( Domain* pD );
509         virtual void RectTouch( Domain *pD );
510         virtual void RectOverlap( Domain* pD);
511         virtual void RectEnd( Domain* pD );
512     }
513
514     main() {
515         510 MyReceiver *pMyReceiver = new MyReceiver;
516         GST gst;
517         gst.SetCell( some Cell);
518         512 gst.Register( CB_TOUCH, pMyReceiver );
519         gst.Register( CB_OVERLAP, pMyReceiver );
520         gst.Traverse
    
```

**FIG. 5A**

```

518 Void MyReceiver::RectTouch( Domain* pD )
{
    {
        if ( pD )
        {
            a = pD ->GetRect();
            b = pD ->GetLayerName();
            .....
        }
    }
}
520

```

**FIG. 5B**



9/15

enum CallbackType [ CB\_FIRST, CB\_TOUCH, CB\_OVERLAP, CB\_END ];

class Receiver

{

public:

virtual ~ Receiver();

virtual void RectFirstEncounter(Domain\*) = 0;

virtual void RectTouch(Domain\*) = 0;

virtual void RectOverlap(Domain\*) = 0;

virtual void RectEnd(Domain\*) = 0;

};

602

604

FIG. 6A

class GST

{

public;

GST();

~GST();

bool Traverse();

void Register(CallBackType, Receiver\*);

void UnRegister(CallBackType);

}

606

608

FIG. 6B

10/15

**FIG. 6C**

```
class GST_Callback
{
private:
    static int Register(CallBackType, GST*);
    static void UnRegister(CallBackType);
private:
    static table gst_registration_table;
friend class GST;
};
```

610

612

614

**FIG. 6D**

```
class GST_Touch : public GST
{
public:
    GST_Touch(Receiver*);
    ~GST_Touch();
    virtual void Execute( Domain*);
private:
    Receiver* pReceiver;
};
```

11/15

```
int GST_CallBack::Register( CallBackType aCB, GST*, pGST )
{
    if ( gst_registration_table.insert(make_pair(aCB, pGST)).second )
        return 1;
    else
    {
        cout<<"GST_CallBack::Register Error: Key already exists" << endl;
        return -1;
    }
}
```

**FIG. 6E**

```
void GST_CallBack::UnRegister(CallBackType aCB )
{
    gst_registration_table.erase( aCB );
}
```

**FIG. 6F**

```
void GST_Touch::Execute( Domain* pD )
{
    pReceiver ->RectTouch( pD );
}
```

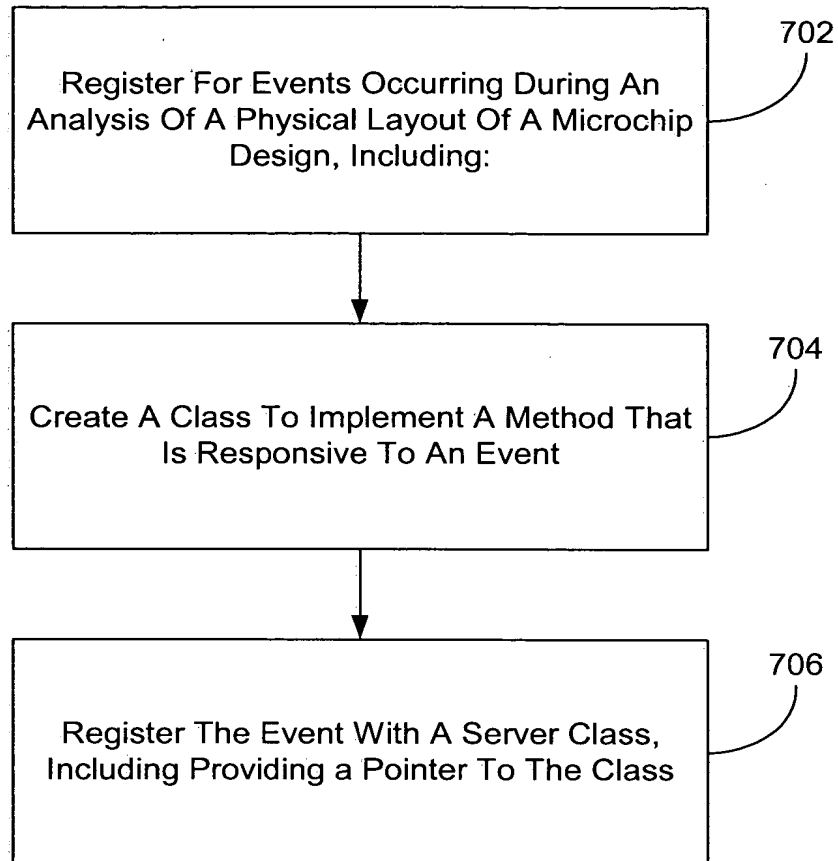
**FIG. 6G**

```
GST::Traverse()
{
    .....
    pTouch = GST_CallBack::gst_registration_table.find( CB_TOUCH )->second;
    .....
    pTouch ->Execute( pDomain );
}
```

**FIG. 6H**

12/15

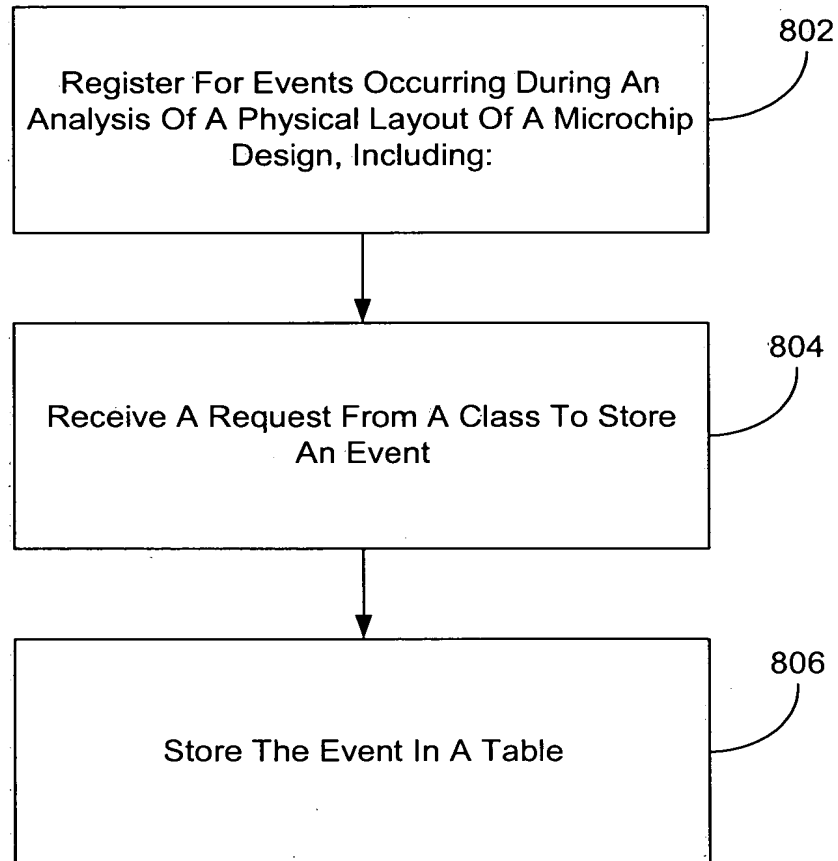
Client Perspective



**FIG. 7**

13/15

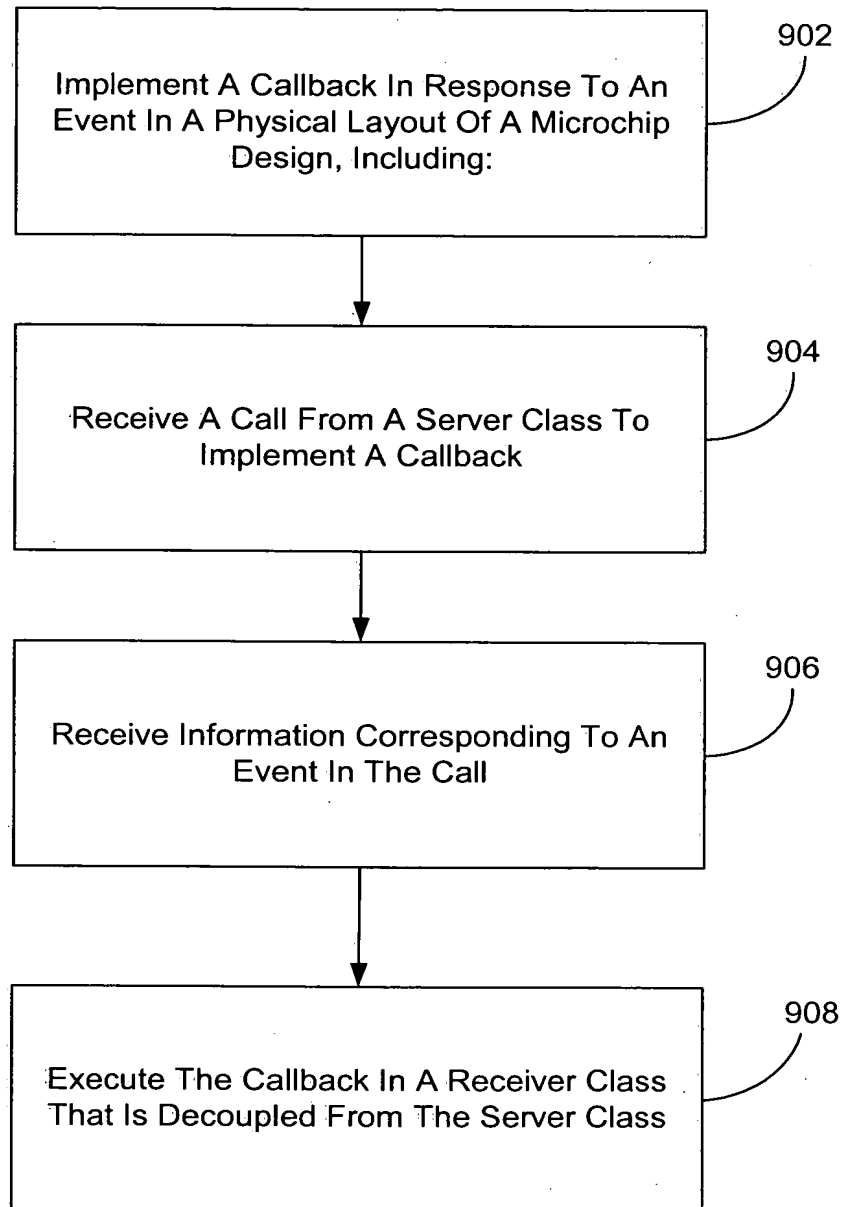
Server Perspective



**FIG. 8**

14/15

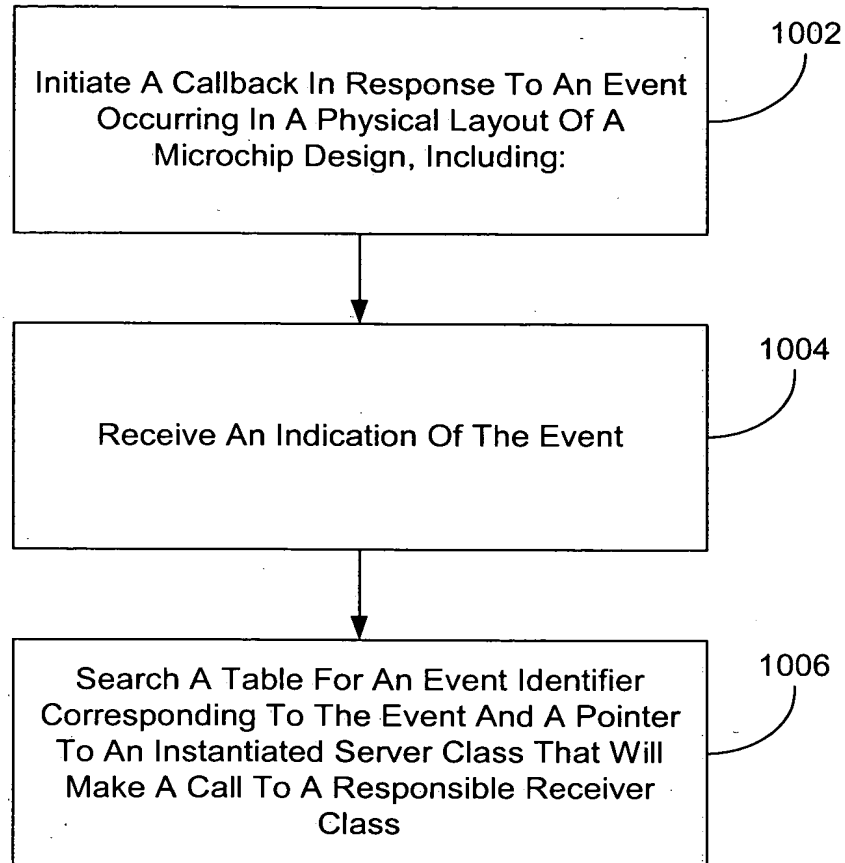
Client Perspective



**FIG. 9**

15/15

Server Perspective



**FIG. 10**